University of Cyprus
Department of Electrical and Computer Engineering

Enhancing the 3D Reconstruction Quality of Natural Scenes: A Pipeline for Evaluation

Author:
Christos Partasidis

Supervisors:
Dr. Lucas Pinto Teixeira
Prof. Dr. Margarita Chli

Committee Members:
Prof. Dr. Margarita Chli
Prof. Dr. Themistoklis Charalambous

A Bachelor's Thesis for the Bachelor's Degree in Computer Engineering.
December 22, 2023

Abstract

In recent years, there have been notable advancements in the field of 3D reconstruction, particularly in the reconstruction of point clouds representing scenes. Scientists have been actively exploring various methodologies to achieve this type of reconstruction. One method is Structure from Motion (SFM) [1] and is implemented in COLMAP [2], the software employed in this research pipeline. In this research work, we aim to assess the performance of COLMAP by employing it in the pipeline with the addition of a digitally created virtual scene that simulates a natural environment. The distinctive aspect of this digital scene lies in its ability to establish a ground truth representation, which is often unattainable in real-world scenes due to factors such as the movement of tree branches and the complex, distinctive characteristics of the trees. Through this evaluation pipeline, we can conduct experiments aimed at identifying specific factors and processes that require consideration, such as the number of images, resolution, and the flight path of a drone that captures the images. These considerations are crucial for enhancing the quality and efficiency of the 3D structure reconstruction in natural environments. Additionally, the scientific community has recognized the emergence of new approaches known as the "learning-based approaches". In these approaches, machine learning models are trained to predict 3D models. A critical component when training these models involves evaluating the predicted results, and here our evaluation pipeline can lay the groundwork for the implementation of new learning-based approaches specifically for natural environments.

Acknowledgments

First, I want to thank the person that allowed to me enter this new and amazing journey, Professor Margarita Chli, who believed in me and gave me the opportunity to take on this project and for her valuable mentoring through the journey. Next up, is the person that we have been in a closer contact, through countless meetings, who he has offered me the right guidance, both in practices for the project technicalities but also for how to become a better engineer and a more mature individual, and that is giving my thanks to Dr. Lucas Pinto Texeira for his enormous help during this project's journey. Finally I want to thank my family and friends for believing in me and allowing me to continue this journey while helping me daily and giving me the mental support to keep moving forward.

Acronyms

Below is a list of the acronyms used in this document:

- **SFM:** Structure From Motion
- **COLMAP:** Collaborative Mapping and Photogrammetry
- **BPA:** Ball Pivoting Algorithm
- **CAD:** Computer Aided Design
- **ML:** Machine Learning
- **CD:** Chamfer Distance
- **EMD:** Earth Mover's Distance
- **JSD:** Jensen-Shannon Divergence
- **PCD:** Point Cloud Data
- **COV:** Coverage
- **MMD:** Minimum Matching Distance
- **PSNR:** Peak Signal-To-Noise Ratio
- **MSE:** Mean Squared Error
- **SSIM:** Structural Similarity Index Measure
- **HVS:** Human Vision System
- **LPIPS:** Learned Perceptual Image Patch Similarity
- **GT:** Ground Truth
- **RMSE:** Root Mean Square Error
- **AABB:** Axis Aligned Bounding Box
- **BB:** Bounding Box
- **VG:** Voxel Grid
- **CO:** Cropped Object

Table of contents

Chapter 1: Introduction

Geometric representation of objects and or scenes is an important task in the field of robotics, computer vision, medicine, and augmented/virtual reality. This task is important in the aforementioned fields because it allows them to preserve the scene, to identify objects within it and classify them in order to perform informed decisions with precision and accuracy. For many years scientists have been developing methods to increase the quality of the reconstruction of geometric structures given inputs of images representing objects and or scenes. Starting from classical methods such as [1] [2] to more recent approaches that utilize machine learning models such as [3],[4], [5] and [6]. Although in recent years an incredible amount of advancement has been foreseen within the scientific community, there has been a lack of assurance and satisfying quality in the reconstruction of natural environments. Before continuing, let's answer the question that some readers might have. Why should we care about the reconstruction of objects and scenes? and more specifically, why should we care about the reconstruction of natural environments? Maintaining precise digital versions of objects and scenes could facilitate numerous applications, enhancing their effectiveness or even making their deployment feasible. Take for example, an emergency drone that navigates through the city. For the emergency drone to be able to navigate within the city's roads it must be able to have an accurate representation of the city such as buildings, cars, and trees. If at any given point of time the drone is unable to detect an object within the scene or construct the scene itself it takes the risk of crashing into other objects such as trees, walls, and poles leading to life threading consequences. To generalize this, for a robot to be able to execute its actions with confidence and take informed decisions to successfully achieve its set goal it needs to be aware of its surroundings. This is why the geometric reconstruction of objects and scenes is important. In the context of natural environments, there are numerous instances where applications require the reconstruction of elements like trees and plants, to make them possibly applied and used. The primary field that requires accurate reconstruction of natural environments is in agriculture. Projections estimate that by 2030 the food production industry will need to double its production lines. In contrast, each year, food production companies face increasing challenges in securing human capital for the repetitive and tough tasks such as weeding, spraying, and fruit picking. This is where robots deployed in agriculture play a key role. An example is a robot designed for autonomous crop management. An essential task of such a robot is to monitor and reconstruct its environment as well as accurately classify the objects surrounding it. This enables it to make informed decisions, such as when to spray, prune, or alert the farmer. Another application within agriculture is the deployment of robots to perform precise picking of fruits and vegetables from the fields. This indeed requires a far more accurate and precise reconstruction of the environment because of the nature of the picking that must be very precise. We have identified that an accurate and precise reconstruction pipeline from input images for natural environments does not exist. In this work we are providing an evaluation pipeline that can be used to evaluate the quality of different methods and is intended to be used as the basis for another pipeline to improve the reconstruction of natural environments. This pipeline implementation includes visualizations that will help the user identify which regions from the specified objects are poorly reconstructed, provides metrics that can help identify the quality of the estimation, includes the basis to be utilized within an extended pipeline to improve the quality and finally utilizes a digital twin to create natural environments that can assist in the process of acquiring large data for the training of a learning based approach.

Chapter 2: Background

2.1 Structure from Motion [1]
It is the process of creating (reconstructing) a 3D estimated model of the scene depicted in the 2D input images that are given. In addition, to improve the quality of the reconstruction camera parameters along with the images are usually provided. They are some standards steps followed in this process. First, a feature detection step takes place to distinguish features on each image. There are various methods used to extract features such as [7], [8], and [9]. Then a feature matching step is performed to try and match images by matching their corresponding features. Afterwards, an estimation of the camera poses takes place (position, orientation) based on the movement of the features. Later, triangulation takes place to estimate the 3D position of those features. In order to reduce the re-projection errors from the 3D points back the 2D images the estimated camera poses, and the estimated 3D point positions are optimized and refined using the Bundle Adjustment method [10]. Finally, the reconstruction creates the 3D model. The output of this process is a 3D point cloud representing the scene from the images and the predicted camera poses of the images.

2.2 Stereo Vision [11]
It is the process to calculate for each pixel in the images their corresponding disparities [12]. Using the disparities, a depth map can be calculated for an image.

2.3 Ball Pivoting Algorithm (BPA) [13]
This is the process that given a point cloud of a scene as input it tries to create a watertight (without holes) surface mesh. The ball pivoting algorithm could be thought of as a ball trying to drop through a point cloud, and whenever it is able to do that while touching 3 points without other points intersecting it creates a triangle for those three points. In our application the ball would represent a drone, and this would have had many advantages. This is why this method would have been the preferable choice. Unfortunately, this method can’t be used because when a diameter of 0.5 meters (m) or more is used which is a real estimation of a drone’s width, it creates a lot of holes on the mesh. Not having a watertight mesh is not useful. Filters were tried to fill in the holes but there were no sufficient results.

2.4 Poisson [14]
It is another approach to try and reconstruct a surface given a point cloud. The drawback is that it loses the application of the ball as in BPA [13]. This approach is still not sufficient because floating parts were present in the experiments performed, which again makes the mesh not watertight.

2.5 Voxelization [15]
The clear advantage of this method is that it is easy to calculate and most of the times it will result to a watertight model. Also, the voxels could be an alternative of the ball, and could have a similar application. The drawbacks are that it loses the exact application of the ball and a voxel size must be chosen.

Chapter 3: Related Works

For many years scientists in the computer vision field have had a long journey in trying to create geometrical structures of objects and or scenes from given input images. A variety of methods have been developed to geometrically represent objects and scenes. They can be categorized into two major categories in terms of their approach. The first category is the classical category where they use techniques such as Structure from Motion [1] and Stereo Vision [11]**.** The second category is the learning-based category where machine learning models such as fully connected neural networks [16] are used to predict the 3D structures. In this category the efforts are to train the models to predict the output. Moreover, each of these categories has a variety of approaches that can be followed. The classical methods are older approaches that are based on years of research and are quite standardized. They receive input of images and then the pipeline constructs a point cloud [17] that would depict the scene present in the images. Afterwards, a mesh [18] can be reconstructed using the point cloud as input with methods such as BPA [13] or Poisson [14]. The learning-based methods also have various pathways to follow. Here they are categorized in three categories based on their output and evaluation method. The first category are those who output point clouds such as [3] and [4], the second are those who predict new views and images [5] and the third categories those who predict meshes [6]. Afterwards, when the training is finished it is presented with several images and it makes the prediction. In order to train the models, they must be presented with the corresponding target output so that the model can learn. The input/output data are either 2D images or 3D models in the form of point clouds. Some approaches try to limit the number of images to a single image [3]. This approach and others mentioned below try to avoid the requirement of having both the input and ground truth models to train the model and try to implement supplementary pipelines to allow them to do the training using other more simple structures e.g. 2D images. Understanding the scene from an input of 2D images has been a problem for many years and thus many approaches/pipelines/methods have been proposed. This is why creating, maintaining, and updating datasets has a key role. The datasets have two main use cases. First, they can be used as a training resource to train new neural networks or other machine learning models to implement methods. Second, they can be used as a benchmark dataset for all the developed methods to evaluate their quality. Most of the datasets include models of objects and or scenes with their corresponding images that depict them. The ground truth model is usually provided after a laser scanner [19] has been used to scan the object or scene. There are also methods to set the ground truth such as using GPS and IMU data to set the ground truth poses, Ground Truth Control Points which are set known points on the Earth's surface, and most interestingly simulated data. Recent advancements in 3D computer aided design (CAD) models that allow the simulation of objects and scenes can contribute significantly to improving these datasets because of the unlimited digital data that could be generated and also their precise corresponding ground truth.

3.1 Datasets

An example of a dataset is Brandenburg Gate provided by OpenHeritage3d [20]. It contains a lot of unordered image collections and a ground truth point cloud depicting the Brandeburg Gate created by LiDAR technology and photogrammetry [21]. Another dataset is the DeepVoxels Dataset [22] that contains four Lambertian objects [23]. These objects are synthetic meaning that they are created using a CAD. The last example is the ShapeNet Dataset [24], which is a large-scale annotated dataset of a variety of different 3D shapes. As of now it covers 55 common object categories with each having about 51,300 unique 3D models but it is constantly updated. What is different about this specific dataset is that it provides 3D models with annotations in large scale. The annotations include the object's basic category, object size, functional parts, surface material, weight etc.

3.2 Classical Methods

One of the classical methods that we have used throughout the thesis to reconstruct different scenes is COLMAP [1] [2]. COLMAP is a classical method introduced in 2016. Specifically, the program was published in 2016 but the techniques it utilizes could be of 30 years of research. Although it is considered old it still competes against state-of-the-art approaches as the basis, but new approaches have shown better performances. Although new approaches have shown better performances many of them have the disadvantage that they must be trained (because they used a learning-based approach) and that are not generally applicable e.g. only single objects, only a specific type of scene, no illumination etc. This is why COLMAP is still relative because it is very generic and can reconstruct many objects and or scenes with high precision. COLMAP uses an SFM [1] process to reconstruct 3d structures of a scene given unordered image collections of a scene as input. The input of this pipeline are images that depict a scene taken from different viewpoints. Optionally, the user can provide camera parameters for a more accurate reconstruction. Finally, the user can create a surface by using methods such as Poisson [14].

3.3 Learning Based Methods

In [3] the proposed method tries to predict a point cloud that will depict the object shown in the single input image. They mention that point clouds are a simple representation of objects and are easy for the machine learning (ML) models to learn because of their simple and uniform structure as well as that they do not require any connectivity combinations. They model the problem's uncertainty as a probability distribution P(.|I) where "I" is the single input image. During training they provide a single sample of P(.|I) for each image. In order for the ML model to learn the predicted output must be compared with the ground truth. To achieve this, a comparison method must be used. Consider two point clouds P1 = $\{x_i \text{ e } R^3\}_{i=1 \text{ to n}}$, and P2 = $\{x_j \text{ e } R^3\}_{j=1 \text{ to m}}$, and let P1 be the predicted (estimated, reconstructed) and P2 the ground truth (target). The authors use two comparison metrics, the first is **Chamfer Distance (CD)** [25], a summation of the distances between points from P1 and their corresponding nearest neighbor in P2 and vice versa. The equation to calculate the metric is:

$$CD(P1, \text{P2}) = \frac{1}{2n}\sum_{i=1}^{n} |x_i - NN(x_i, \text{P1})| + \frac{1}{2m}\sum_{j=1}^{n} |x_j - NN(x_j, \text{P2})| \qquad \text{(eq. 1)}$$

where NN(x,P) is a function that finds the nearest neighbor of point x within the point set P. The lower the CD the better the quality of the predicted point cloud. Thus, having a lower CD is desired. The second metric is **Earth Mover's Distance (EMD)** [26] and the equation to calculate it is:

$$EMD\ (P1, P2) = \min_{\pi \in \Pi(P1,P2)} \sum_{i=1}^{n}\sum_{j=1}^{m} \pi_{i,j} |p1_i - p2_j| \qquad \text{(eq. 2)}$$

where π (P1,P2) is the set of n x m matrices where the rows and columns sum to one [26]. The lower the EMD the better the quality of the predicted point cloud. Thus, having a lower EMD is desired. Also, these metrics are used for the evaluation of their proposed approach. In [4] a neural network is proposed to generate new point clouds using object models provided as input during the training phase. The object models are in PCD format and are from known datasets such as ShapeNet [24] that includes several types of objects. The authors use CD and EMD. Additionally, it mentions three new metrics from [27]. The first metric is the **Jensen-Shannon Divergence (JSD)** [27] metric. It is used to evaluate the similarity between two voxel grids. Specifically, it is performed on the point cloud data (PCD) of the voxel grids. Assume a point set A and a point set B. It evaluates the occupied space from points of a set A compared to a set B. More specifically, it counts the number of points within each voxel for both voxel grids (A and B) creating two distributions P_A and P_B and then computes the JSD calculated by the following equation:

$$JSD(P_A \parallel P_B) = \frac{1}{2} D(P_A \parallel M) + \frac{1}{2} D(P_B \parallel M) \qquad \text{(eq. 3)}$$

where M = ½ (P_A + P_B) and D(. || .) is the KL-divergence between the two distributions [28]. The lower the JSD the better the quality of the predicted point cloud. Thus, having a lower JSD is desired. The second metric is **Coverage (COV)** [27]. This metric will calculate how many points have been covered in one point set compared to another. Assume a point set P1 and a point set P2. Now, for each point in P1 we find the nearest neighbor in P2 and mark each as a match. The closest point can be found using the CD or EMD. The metric is the fraction of the points that were matched in the point set P2. The equation is given below:

$$COV(\mathrm{P1}, \mathrm{P2}) = \frac{\left|\{\arg min_{\mathrm{y}\in \mathrm{P2}} \mathrm{D}(\mathrm{x}, \mathrm{y}) \,|\, \mathrm{x} \in \mathrm{P1}\}\right|}{|\mathrm{P2}|} \quad \text{(eq. 4)}$$

where x, y are points in the point sets P1 and P2 correspondingly, and D(.,.) can be either CD or EMD. Also, the authors use the above metric by having P1 as the predicted/reconstructed (generated) and P2 the ground truth (reference) point clouds. To conclude, the higher the COV the better the quality of the predicted point cloud. Thus, having a higher COV is desired. The third and last metric is the **Minimum Matching Distance (MMD)** [27]. The coverage metric COV has the drawback that it does not show how close the closest point is. This is what the MMD tries to improve. It utilizes either CD or EMD to calculate the distances. Now, for each point in P2 the distance to the nearest neighbor in P1 is calculated. The distances are added and then they are averaged by dividing with the total number of points in P2. The equation can be seen below:

$$MMD(P1, P2) = \frac{1}{|P2|} \sum_{y \in P2} min_{x \in P1}\ D(x, y) \quad \text{(eq. 5)}$$

where x, y are points in the point sets P1 and P2 correspondingly, and D(.,.) can be either CD [25] or EMD [26]. To conclude, the lower the MMD the better the quality of the predicted point cloud. Thus, having a lower MMD is desired. In Nerf [5], a novel approach for predicting new views of objects and scenes by training machine learning models is proposed. The major difference between the aforementioned approaches is that this proposal tries an approach to eliminate the need of having a 3D model of the ground truth scene and instead uses only 2D images for the evaluation criteria. In this way it can reduce computational load and storage drastically. Also, obtaining 3D ground truth of a scene or object is cumbersome although with the recent advancements in CADs this is easier. The input is a set of images with known camera poses. Moreover, the output is new views that can later be used to create the model of the 3D scene. Specifically, they represent a scene with the use of an multilayer perceptron network [29]. That means that the neural network in simple terms becomes the scene. To get the generated/predicted images, they implement a method to hierarchically and structurally sample images from different viewpoints, and then they compare them with the corresponding ground truth images. All the comparison metrics evaluate the similarities between images. Starting from **Peak Signal-To-Noise Ratio (PSNR)** [30] which is the simplest metric out of the three, it first calculates the MSE [31] between the original and the reconstructed image by calculating the squared difference of each pixel from the two images and then calculates the average. Assuming images with m x n dimensions, the equation for the MSE between two images is:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \quad \text{(eq. 6)}$$

where "I" is the noise-free image (ground truth) and "K" is the noisy approximation (reconstructed). Then the maximum peak value must be defined denoted as "MAX_I". Here the pixels take a maximum value of 255 so the maximum peak is that. Finally, to calculate the PNSR the equation is:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE}\right) \quad \text{(eq. 7)}$$

Note that for pixels with RGB values the MSE is calculated three times for each red, green, blue part and then an average of these three is calculated. To conclude, the higher the PSNR the better the quality of the distorted image. Thus, having a higher PSNR is desired. The second metric is **Structural Similarity Index Measure (SSIM)** [32]. It is based on the perception of the Human Vision System (HVS), and considers three different factors. The factors are changes in the structure, illumination, and contrast of the images. The equation for two images x and y of size N x N pixels is:

$$SSIM\ (x,y) = \frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)} \quad \text{(eq. 8)}$$

where details can be found in [32]. The factors are computed in different windows on the image. To conclude, the higher the SSIM metric the better the quality of the distorted image. Thus, having a higher SSIM metric is desired. The third metric, that is the most advanced and interesting is the **Learned Perceptual Image Patch Similarity (LPIPS)** [33]. It utilizes neural networks to output a metric. The learning and evaluation of the neural networks is performed using large datasets created previously by other scientists such as LIVE [34], TID2008 [35] and their own. These large datasets are created by querying people on differences between images with different distortions, and at different patches of the images. After the datasets are complete, they can be used for training the neural networks. To conclude, the lower the LPIPS metric the better the quality of the distorted image. Thus, having a lower LPIPS metric is desired. Furthermore, reconstructing a single object and reconstructing a whole scene are two completely different approaches. In [6] a novel approach was proposed to learn characteristics of different objects and then combine them to compose the full scene. Specifically, their implementation takes as input a sparse set of point cloud and reconstructs the scene creating a watertight mesh. The metrics used to evaluate their approach are CD [25], and **Normal Alignment** [36] which is the mean absolute dot product of the normals in one mesh and the normals at the corresponding nearest neighbors in the other mesh.

3.4 Summary of Selected Works

COLMAP performs great in general, but it lacks quality in reconstructing trees because of their complex and detailed structure as well as the existence of movement in the branches of the trees. Now, all of the aforementioned learning-based approaches perform very impressive results but none of them have reported any details about their performance on trees and natural environments. This leaves the potential for improvement, to build a model that is trained to predict natural scenes. A critical step in the training is the evaluation of the predicted model at each iteration step. Three categories of evaluation metrics were presented. The first category performs a comparison between point clouds and on points of voxel grids such as CD [25], EMD [26], JSD [27], COV [27], and MMD [27]. The second category compares images with PSNR [30], SSIM[32] , and LPIPS [33] and the third and last category compares meshes [36].

3.5 Related Works vs Our Proposal

Considering the simplicity in the structure of voxel grids, coupled with their increased probability of successfully generating a predicted, watertight structure (without holes), we have decided to focus our analysis on voxel grids. This choice is made with the objective of developing a pipeline that will be used in a learning-based approach to predict voxel grids. This pipeline compares voxel grids and their corresponding points. We have implemented a different variation of Coverage [27] and CD [25] that are adapted for voxel grids explained in Chapter 4.

Chapter 4: Methodology

The proposed work implements an evaluation pipeline that can be used to evaluate reconstruction methods but also is aimed to be extended within a learning-based approach. The pipeline offers:

- **Cropping Objects:** Performs a cropping process to extract the specified objects from a scene, so they can be evaluated and used for a learning-based approach.
- **Visualizations:** It provides visualizations for the cropped object's coverage and distance.
- **Evaluation metrics:** Different metrics are provided to evaluate the estimated cropped objects and estimated image positions.
- **Digital Twin:** We are using a CAD system to generate natural scenes, enabling us to create numerous scenes along with their respective ground truths as needed.

In summary, it consists of 9 steps. These steps are to evaluate the classical method COLMAP [2] but similarly this pipeline can be integrated within a learning-based approach. The evaluation compares the estimated cropped objects with the ground truth cropped objects.

4.1 Step 1: Create Digital Scene

Starting with the creation of the digital scene. This is a crucial step as it enables us to digitally generate an unlimited number of natural environments. These scenes can then be used to train a learning-based model. The inputs of this step are the different models of objects that compose the scene such as trees, grass, fences, terrain, sky etc. with their corresponding textures, and the configurations that are initially provided by the user. These configurations could be the size of the field, the number of trees, the view of the capturing images and other. Then the generated digital scene is created, and images are captured depicting the scene. The output is images depicting the scene. An example of images depicting a natural scene that contains four winter trees, with grass and fences in a flat terrain captured using an “aerial around” view mode can be seen in Figure 1.

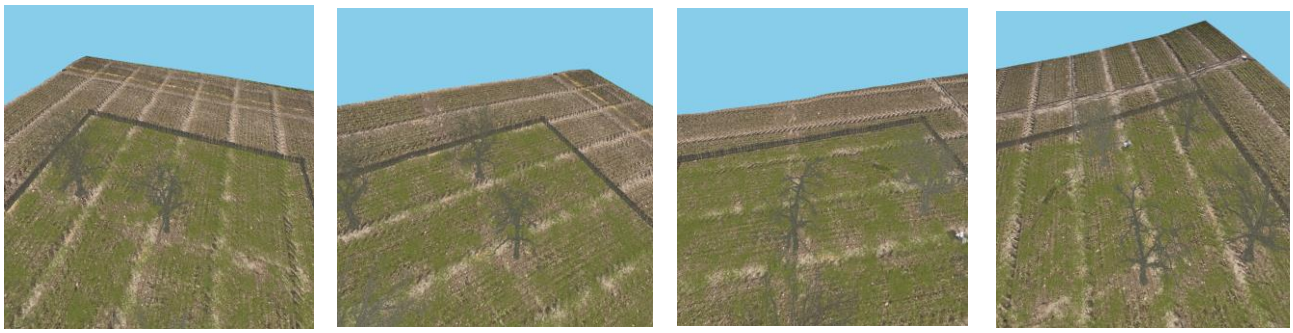


Figure 1: Images of the digital scene with winter trees captured with around view

4.2 Step 2: Sparse Reconstruction

Using COLMAP [2], we will be creating a sparse point cloud that depicts the scene in the input images. The input of this step are the images retrieved from Step 1. COLMAP [2] using SFM [1] reconstructs the estimated scene in the form of a point cloud, and it also estimates the image poses.

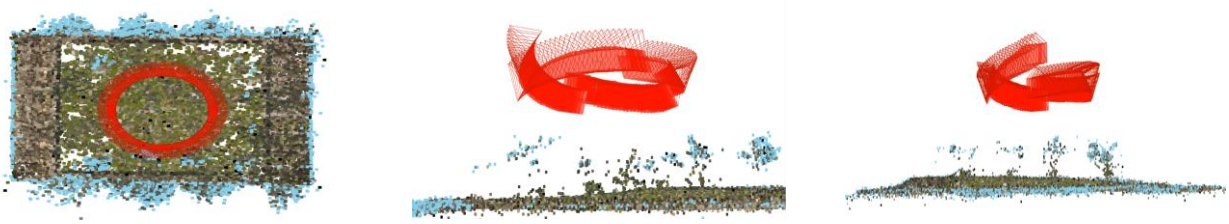


Figure 2: Sparse reconstruction using images captured with around view

In Figure 2 the red markers represent the estimated image poses and underneath them are the estimated points that depict the scene seen in the images.

4.3 Step 3: Evaluate Positions

An evaluation of the estimated image positions takes place. The inputs of this step are the ground truth (GT) image positions and the estimated (COLMAP) image positions. First, an alignment by minimizing the distances between the two trajectories of the image positions happens and then the error is calculated using the Root Mean Square Error (RMSE) [37] between corresponding image positions. The outputs are visualizations of the two trajectories, and the error between them in meters (m). In Figure 3 below is an visualization example of the trajectories of the image positions.

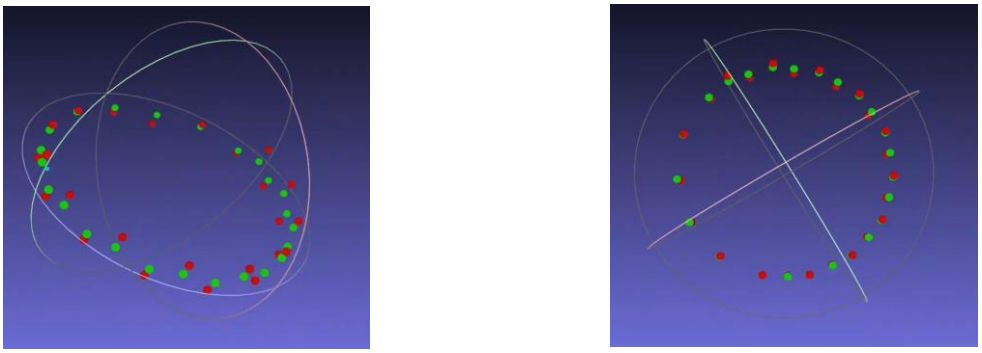


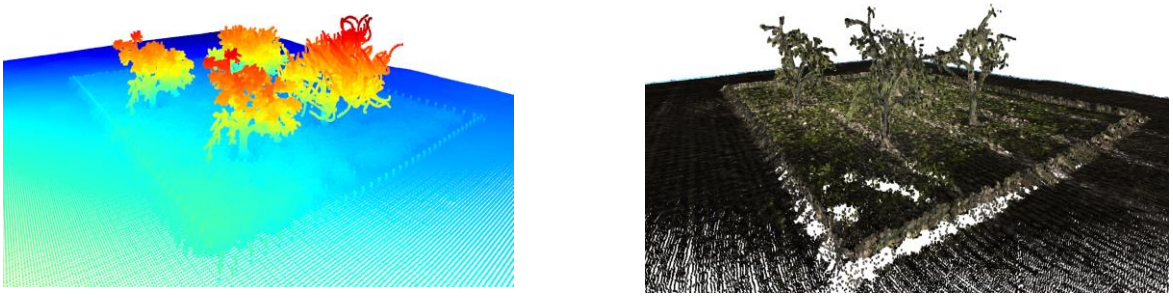
Figure 3: Image positions trajectories. (Green) GT. (RED) Estimated.

4.4 Step 4: Create Ground Truth Scene

Afterwards, the ground truth scene must be created to compare it with the estimated reconstructed scene. The inputs of this step are the object models and their corresponding poses which were created during Step 1. The process combines the objects into one scene. It places each object at the specified pose. The output is the ground truth point cloud.

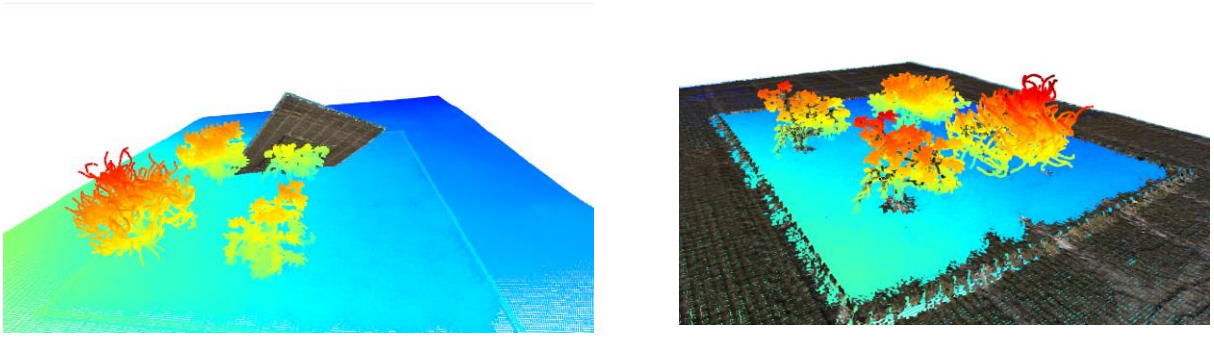
4.5 Step 5: Align Models

Before cropping the objects an alignment between the GT and the estimated point clouds must be performed. This is crucial because in Step 6 where the cropping of the objects is performed we have to crop the right areas. The inputs of this step are the GT image poses, the estimated PCD and the estimated image poses. The process aligns the estimated PCD based on the estimated image poses. A transformation between the two trajectories of the image poses that minimizes the error between them is estimated, and then applies that transformation on the estimated PCD. The output is the aligned estimated PCD. In Figure 4 are PCDs and in Figure 5 the before and after of the alignment.



(a) (b)

Figure 4: PCD. (a) GT. (b) Estimated/COLMAP.



(a) (b)

Figure 5: Both GT and COLMAP PCD. (a) Before Alignment. (b) After Alignment.

4.6 Step 6: Dense Reconstruction

A denser point cloud is created using the aligned sparse reconstruction as input. The output is an aligned dense PCD.

4.7 Step 7: Crop Objects

The cropping step receives as inputs the Estimated/COLMAP and the GT PCDs, crops the selected/specified objects (currently set to winter trees), and stores the cropped objects (in PCD form) for both. The following steps are for a single specified object. Keep in mind that a scene may contain multiple specified objects, for example four trees, so the following steps will be executed multiple times. First, it reads the point cloud of the specified object from the stored object model used in Step 1. Then the axis aligned bounding box (AABB) of that object is calculated and the corner points of it are also calculated.

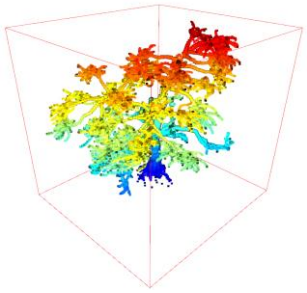


Figure 6: A single object PCD with its corresponding AABB

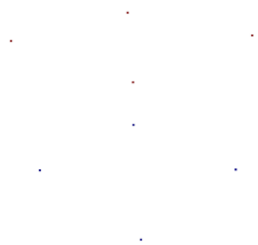


Figure 7: Points of the AABB

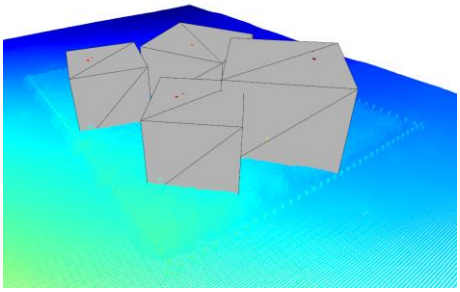
Afterwards, we transform (translate and rotate) the corner points, so the corresponding bounding box (BB) takes the appropriate pose to cover the object in the aligned PCD scenes, so the cropping process later crops the GT and Estimated object correctly. The GT pose of the object is known from Step 1 during the creation of the GT scene. The pose of the estimated object is not known, but because of the alignment process if the method used to estimate the scene is accurate it will crop the right object. Also, note that the cropped object on the GT scene is different from the raw GT object model because the scene also contains other objects. This is preferable because it takes into consideration the overall scene. The transformation is applied on the points of the AABB using the transformation matrices seen below:

$$\begin{bmatrix} a & b & c & T_x \\ d & e & f & T_y \\ g & h & i & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} = \begin{bmatrix} x_f \\ y_f \\ z_f \\ 1 \end{bmatrix} \quad \text{(eq. 9)}$$

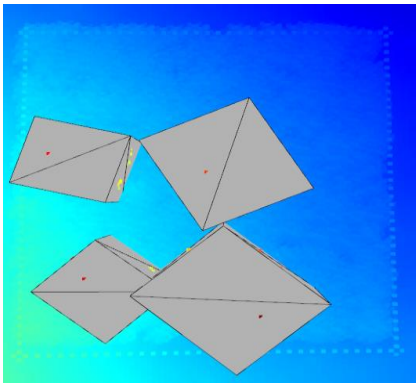
where:

- x_f, y_f, z_f : final coordinates of the transformed point
- x_i, y_i, z_i : initial coordinates of the point before the transformation
- T_x, T_y, T_z: these three values constitute the translation matrix
- a, b, c, d, e, f, g, h, I: all these values constitute the rotation matrix

Then, a mesh of the transformed AABB that is now called a BB is created to be used for the visualization afterwards to check the accuracy of the transformation. In Figure 8 you can see the meshes of the BBs with the GT scene and in Figure 9 with the Estimated scene after this process is performed for all the specified objects.

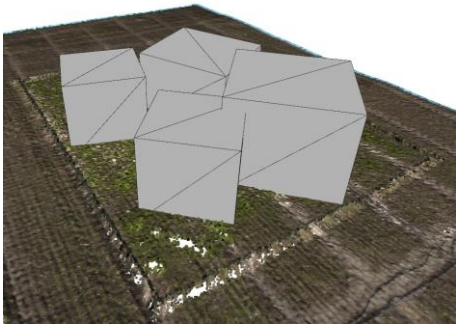


(side view)

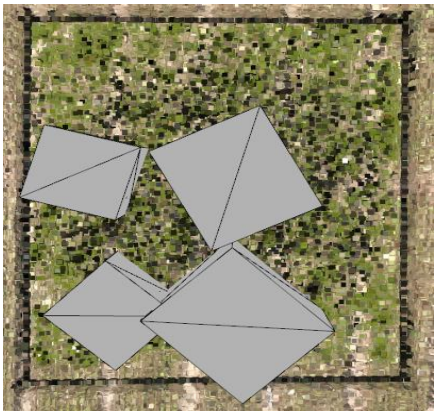


(top view)

Figure 8: GT scene and the meshes of the BBs of the objects



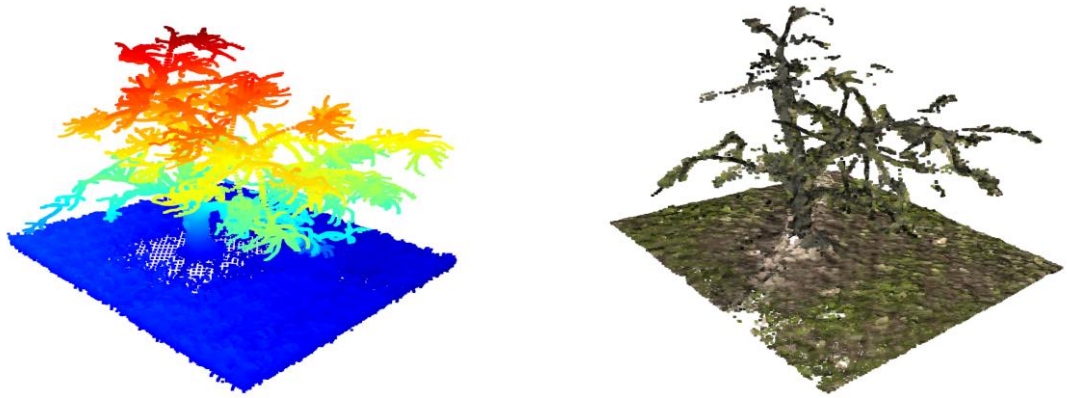
(side view)



(top view)

Figure 9: Estimated scene and the meshes of the BBs of the objects

When the transformation of all the AABBs is finished, the cropping process begins and extracts the specified objects from the two PCDs and stores them in separate directories. The cropping process uses the transformed points of the AABB as bounds to isolate the points from the PCDs.



(a) (b)

Figure 10: A single cropped object. (a) GT. (b) Estimated

4.8 Step 8: Voxelize Objects

In this step we voxelize both the GT and the Estimated/COLMAP cropped objects. It transforms a cropped object from a PCD form to a voxel grid (VG). The voxel grids are assigned colors to distinguish them. We assign the color blue for the GT voxel grid and the color green for the estimated. The voxel size used is 0.3 m meaning that the distance from its center to one of its sides is 0.15 m.

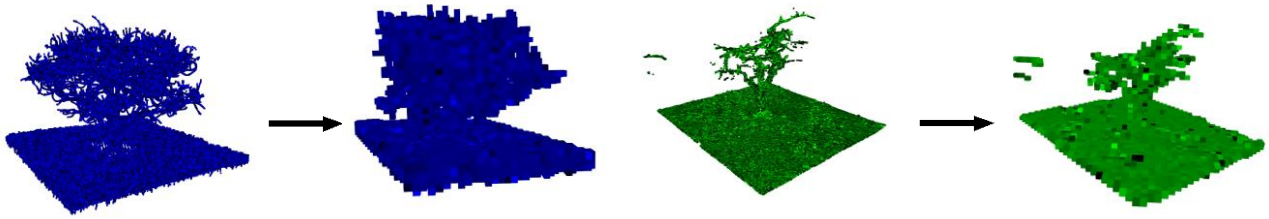


Figure 11: Voxelization from PCD to VG

The reason that we voxelize is because it can reduce the size of points significantly, allowing us to run the pipeline in a feasible amount of time. Remember that a lot of detail is not ideal, we just need enough points that will help reconstruct a watertight mesh. Also, voxel grids are watertight that is why they are desirable.

4.9 Step 9: Evaluate Voxel Grids

4.9.1 Voxel Size and Bound

We have been using a voxel size of 0.3 m and a search bound of 0.3 m. This will lead to having a search space equal to twice the size of the voxel size. Figure 12 is to help you understand the relationship between the voxel size and the bound. Setting up a bound is necessary because the matching would otherwise not be feasible, and the distance error calculation would be wrong. We set the bound to be twice the size of the voxel because we think that it is a good convention, but this must change with different applications, objects, and scenes.

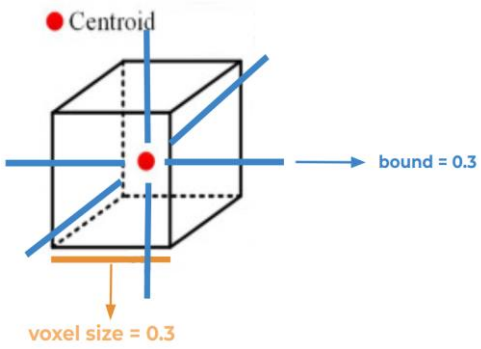


Figure 12: A figure showing the relationship between the voxel size and the bound

4.9.2 Coverage/Matching Metrics

Now we will be evaluating the VGs of the cropped objects. We will be comparing two voxel grids, let's denote them as source and target. The inputs of this step are the cropped GT and COLMAP voxel grids. There are two types of tests being performed. The first test will result to the Coverage/Matching metrics. In this evaluation test, for each voxel in the source VG we try to find a corresponding voxel in the target VG. The search and matching process uses the centers of the voxels and a specified bound to create the search space for each voxel. For each voxel in the source VG, we check whether a voxel's center from the target VG is within its search space. This is checked by the following equations:

$$\begin{aligned} \mathrm{Cs_x} - \mathrm{bound} \leq\ & Ct_x \leq \mathrm{Cs_x} + \mathrm{bound} \\ Cs_y - \mathrm{bound} \leq\ & \mathrm{Ct_y} \leq \mathrm{Cs_y} + \mathrm{bound} \\ \mathrm{Cs_z} - \mathrm{bound} \leq\ & \mathrm{Ct_z} \leq \mathrm{Cs_z} + \mathrm{bound} \end{aligned} \qquad \text{(eq. 10)}$$

where Cs is a center at the source VG, and Ct is a center at the target VG. For a voxel, if any center is found within its search space (all above equations hold), then a corresponding voxel is found within its search space and thus that voxel (of the source) is marked as "found/true". If a center is not found it is marked as "not found/false". Two equations are produced from this. The first equation is:

$$\text{Matching Precision} = \text{True Voxels} / \text{All Source Voxels} \qquad \text{(eq. 11)}$$

The "True Voxels" is the number of voxels matched, and the "All Source Voxels" is the total number of voxels in the source VG (matched or unmatched). This metric shows how many voxels have been correctly matched. The second equation is:

$$\text{Matching Recall} = \text{True Voxels} / \text{All Target Voxels} \qquad \text{(eq. 12)}$$

The "All Target Voxels" is the total number of voxels that exist in the Target VG. The recall here also takes into consideration the total amount of voxels that could have been predicted by using the number of the total voxels in the target VG. In addition, the visualization of this test is the following. If a voxel is not found its color is set to a variation of red, else it retains its original color, green for the Estimated and blue for the GT. In Figure 13 you can see this convention.

4.9.3 Distance Error Metrics

The second test will result to the Distance Error Metrics. This test calculates the minimum matching distance for each voxel from the source VG to the target VG within its search space that is set from the bound. The distance between voxels can be calculated using their centers as their reference points. The distance is calculated by the equation:

$$D = \sqrt{(Cs_x - Ct_x)^2 + \left(Cs_y - Ct_y\right)^2 + (Cs_z - Ct_z)^2} \quad \text{(eq. 13)}$$

where Cs is a center at the source VG, and Ct is a center at the target VG. Specifically, we search for all the centers in the source VG for a corresponding center point in the target VG that has the minimum distance within their search space. We take only into account the distances found that are smaller than the bound (D < bound). If a distance smaller than the bound is not found, then the distance is set to the bound. This allows us to have a maximum boundary that is equal to the bound set for the search space. Then when this process is done for each voxel in the source VG, we calculate the RMSE [37] for all the minimum matching distances of the voxels. This will result in the distance error metric. For the visualization a color encoder is used, where colors are set for each voxel based on their minimum distance. In the experiments to follow the VIRIDIS [38] color encoding is used which has a convention that can be seen in the Figures 14-15. In summary the color is yellow if the distance is high, green if its medium and purple if its low. Remember for better quality a low distance is desired.

Pseudocode:
Initialize a list to store all indicators for matching, from source VG to target VG - (M)
Initialize a list to store all the minimum distances, from the source VG to the target VG - (D)
For each center **in** the source VG:
  Get center
  # Matching Metric
  Set variable that indicates match to **false**
  For each center **in** the target VG:
    If the target center is within the search space:
      Set match variable to **true** and get out of this inner loop
  Append to list M the matching variable (**true** or **false**)
  # Distance Metric
  Initialize a list to store all the distances from this source voxel to all voxels in the target VG – S
  For each center **in** the target VG:
    Calculate distance between source and target voxel
    If distance < bound
      Append distance to the list S
    Else
      Append bound to the list S
  Find minimum distance in S
  Append minimum distance found from S in D
Calculate RMSE for all distances from D

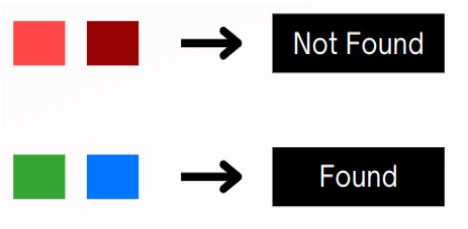


Figure 13: Color Encoding for matching

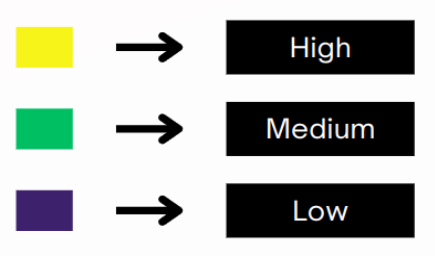


Figure 14: Color Encoding for distances

Figure 15: Viridis color encoder [38]. (Yellow) High. (Green) Medium. (Purple) Low.

The evaluation process takes place twice, one where the source is the COLMAP and the target is the GT denoted as COLMAP → GT, and the other where that the source is the GT, and the target is the COLMAP denoted as GT → COLMAP. Below you can see examples of the visualizations of GT → COLMAP for one cropped object tree. This is a single cropped voxel grid from the GT scene. The Coverage/Matching test is in Figure 16 and for the Distance Error test is in Figure 17.

Figure 16: A single GT cropped object (CO) tree in VG format. (Blue) Found. (Red) Not Found.

Figure 17: A single GT CO tree in VG format using Viridis [38] Color Encoder

Below you can see examples of the visualizations of COLMAP → GT for one cropped object tree. This is a single cropped voxel grid from the COLMAP scene. The Coverage/Matching test is in Figure 18 and for the Distance Error test is in Figure 19.

Figure 18: A single COLMAP CO tree in VG format. (Green) Found. (Red) Not Found.

Figure 19: A single COLMAP CO tree in VG format using Viridis [38] Color Endoder

Chapter 5: Experiments

5.1 Details of Experiments

All the experiments have the following common configurations. The scene includes four winter trees that are about 6-8 (m) tall, with grasses, and fences in the Kastelhof terrain. Also, remember that a scene can include numerous objects that we want to crop. In the experiments below, we are evaluating the trees, so the cropped objects are the trees. The reported Matching Precision and Matching Recall are only the maximum (MAX) values that have been found within the scene, but the Distance Errors are the average (AVG) of all the objects. For the experiments related to Precision and Recall the source is the COLMAP VG and the target is the GT VG. For the experiments related to the Distance Errors it depends on the metric, for the GT→COLMAP metric the source is the GT, and the target is the COLMAP and for COLMAP→GT the opposite. A voxel and a bound size of 0.3 m has been used for all the experiments except for the experiments in 5.5 where the sizes are varied.

5.2 Number of Images

The numbers of images that have been tested are 80, 120, 140, 160 and 200. Below you can see the exact values for the Matching Precision and Recall (in %) for each of the different tests:

Precision = [99.36, 99.52, 99.39, 100, 100]
Recall = [39, 42, 58, 51, 69]

and in Figure 20 a graph that depicts these values. On the x-axis we have the different number of images, and on the y-axis the values for the Precision and the Recall (in %). With red color we present the Precision of COLMAP and the Recall with green color.

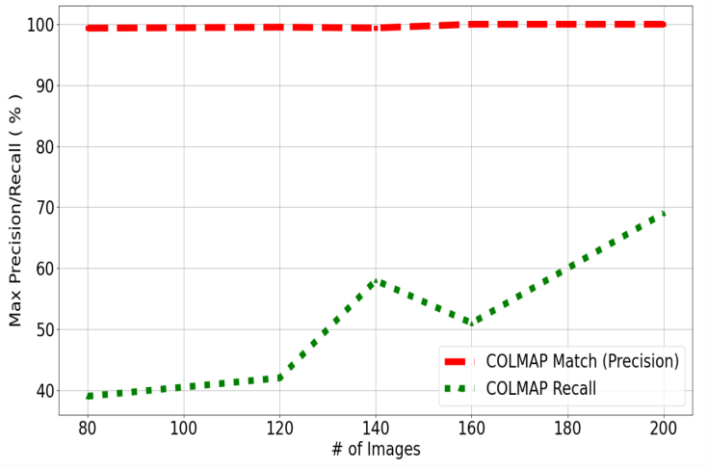


Figure 20: A line graph comparing the Matching Precision and Recall versus the number of images

Now, we also tested how are the Distance Errors affected by the number of images. The values in (m) can be seen below:

colmap_to_gt_avg_rmse =
[0.11, 0.10, 0.08, 0.11, 0.12]
gt_to_colmap_avg_rmse =
[0.24, 0.25, 0.23, 0.24, 0.22]

and in Figure 21 a graph that depicts these values. On the x-axis we have the different number of images, and on the y-axis the values of the Distance Errors for both COLMAP → GT and GT → COLMAP. With red color we present the COLMAP → GT distance error and with blue color the GT → COLMAP.

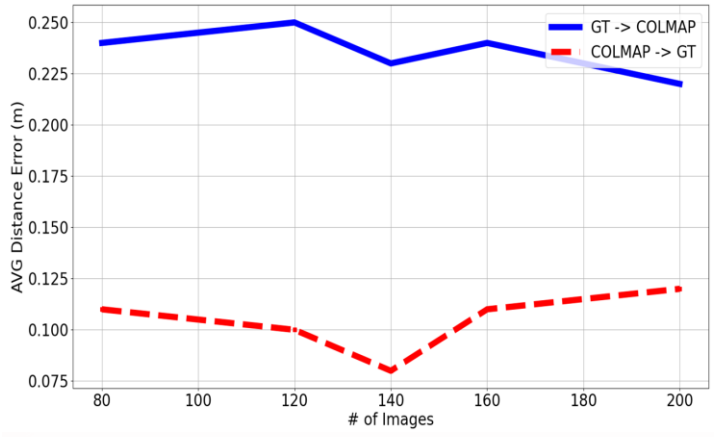


Figure 21: A line graph comparing the Distance Errors versus the number of images

5.3 Resolution

The different resolutions (width x height in pixels) that have been tested are 3072x2304, 2400x1800, 1920x1440, 1440x1080, 1280x960, 720x540 and 480x360. Starting with how the Matching Precision and the Recall (in %) are affected by the resolution below you can see their values for the different tests:

Precision = [72.49, 99.95, 99.88, 99.95, 99.88, 99.95, 100]
Recall = [22, 37, 34, 51, 51, 53, 52]

and in Figure 22 you see the graph that depicts these values. On the x-axis we have the different resolutions (only height), and on the y-axis the values for the Precision and the Recall (in %).

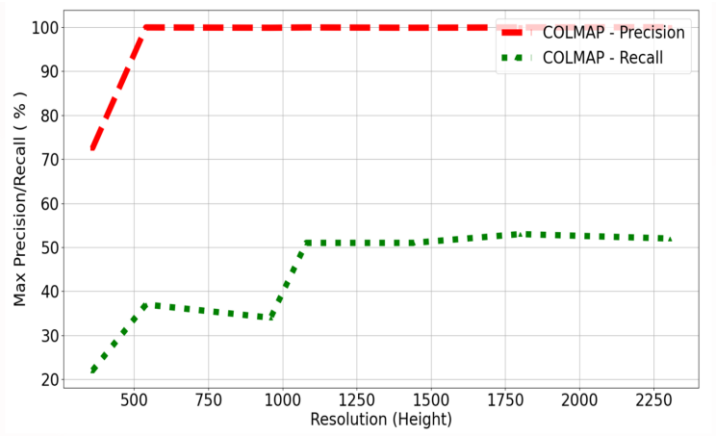


Figure 22: A line graph comparing the Matching Precision and the Recall versus the resolution

Now, we will examine how the Distance Errors are affected from the resolution. Below you can see the values of the distance errors in meters (m) from COLMAP → GT and GT → COLMAP:

colmap_to_gt_avg_rmse =
[0.26, 0.15, 0.11, 0.09,0.11, 0.11, 0.11]
gt_to_colmap_avg_rmse =
[0.29, 0.26, 0.25, 0.23, 0.22, 0.23, 0.22]

and in Figure 23 the graph depicts these values. With red color we depict the COLMAP → GT distance error and with blue color the GT → COLMAP.

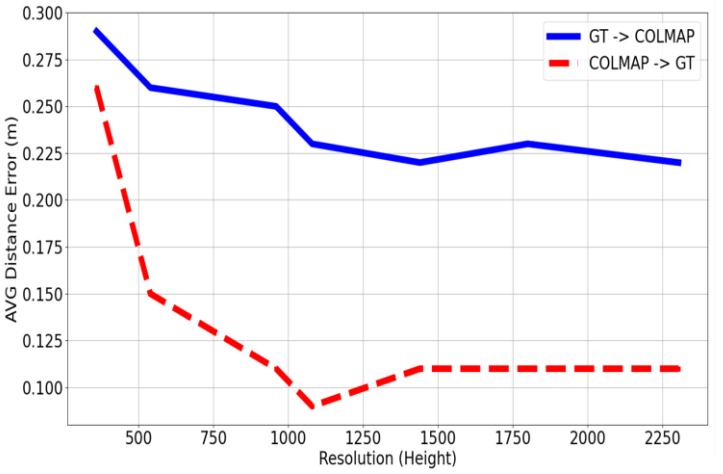


Figure 23: A line graph comparing the Distance Errors versus the resolution

5.4 Degrees added at trajectory

Now in this type of experiment, we are adding degrees at the overall trajectory during Step 1. The default degrees are 360 forming a single circle. The different degrees added to the trajectory at the different tests are 45, 90, 180, 240, 360, 720, 1080, 1440, and 1800. For this experiment a single metric is shown that represents the Trajectory Error in meters (m) calculated by the tool in Step 3. Below you can find the different trajectory errors of each test:

trajectory_rmse =
[2.7, 1.7, 1.6, 1.8, 7.5, 9.0, 8.4, 9.2, 7.6]

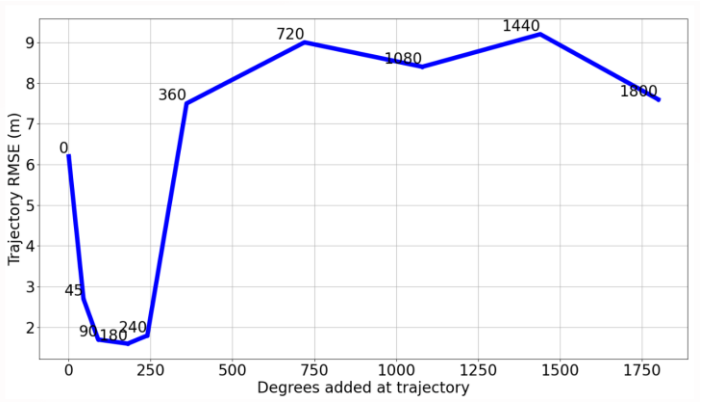


Figure 24: A line graph comparing the Trajectory Error versus the degrees added at the trajectory

In Figure 24 a line graph depicts the relation between the trajectory RMSE and the degrees added. On the x-axis we have the different degrees added, and on the y-axis the values for the Trajectory Error in meters (m).

5.5 Voxel Size and Bound

The different voxel sizes and bounds that have been tested are 0.2, 0.3, 0.4, 0.5, 0.7, 0.9, 1.2, 1.5, 1.7, and 2.0 meters (m). First, we check the affects on Matching Precision and Recall. The values can be seen below:

Precision = [97.87, 99.36, 99.65, 99.97, 100, 100, 100, 100, 100]
Recall = [64, 69, 71, 71, 71, 71, 71, 71, 71, 71]

and in Figure 25 you see the graph that depicts these values. On the x-axis we have Voxel and Bound sizes and on the y-axis the values for the Precision and the Recall (in %).

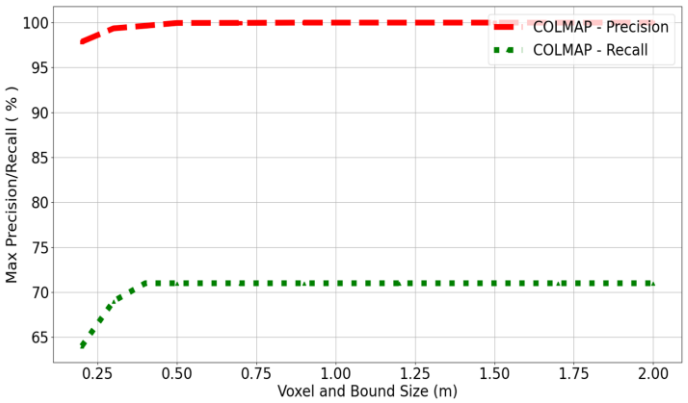


Figure 25: A line graph comparing the Matching Precision and the Recall versus the Voxel and Bound Size

Now, we will examine how the Distance Errors are affected from the Voxel and Bound sizes. Below you can see the values of the distance errors in meters (m) from COLMAP → GT and GT → COLMAP:

colmap_to_gt = [0.11 , 0.12, 0.13, 0.13, 0.13, 0.13, 0.13, 0.13, 0.13, 0.13]
gt_to_colmap = [0.16, 0.22, 0.27, 0.30, 0.36, 0.41, 0.46, 0.50, 0.53, 0.55]

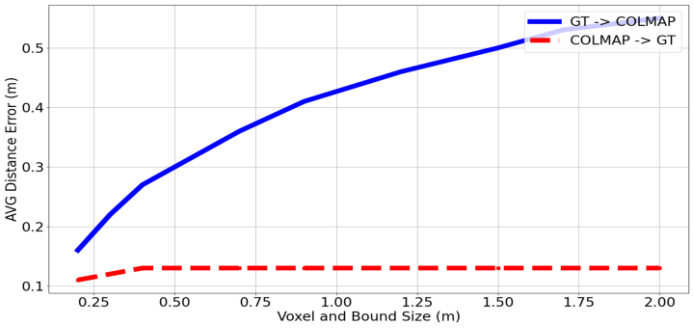


Figure 26: A line graph comparing the Distance Errors versus the Voxel and Bound Size

and in Figure 26 the graph depicts these values. With red color the we depict the COLMAP → GT distance error and with blue color the GT → COLMAP.

Chapter 6: Results

6.1 Number of Images

Precision and Recall

We can observe that the precision of COLMAP is invariant of the number of images used because it is constant at nearly 100%. This does not imply that COLMAP is perfect, but it shows that it will not make any predictions/estimations if it is unsure. This can be seen in the recall of the graph. Starting with a recall of 40%, meaning that the estimated voxel grid that was created is missing more than half of the voxels at ground truth when using 80 images. Now, by more than doubling the images, reaching 200 images the recall is increased by 30% leading to a recall of nearly 70%. This leads to the fact that by increasing the number of images the recall increases but is not proportional.

Distance Error

Regarding the distance errors from both COLMAP → GT and GT → COLMAP, it does not follow a pattern and it roughly remains the same. Although, in some tests there is a small increase and in others a small decrease. This leads to the fact that the error is invariant of the number of images used. Keep in mind that this evaluation applies only for COLMAP. It confirms the fact that COLMAP does not predict unless it is certain.

6.2 Resolution

Precision and Recall

COLMAP's precision is not affected by the resolution of the images except at extreme cases where the image size is small e.g. smaller than 640 pixels height. Regarding the recall, it increases proportionally with the resolution but stabilizes around 1140 pixels height. To conclude, increasing the resolution of the images is beneficial for recall (desired) but increasing the resolution above 1140 pixels height is redundant and can decrease the efficiency significantly.

Distance Error

By increasing the resolution, the distance error from both directions COLMAP → GT and GT → COLMAP decreases significantly especially for the COLMAP → GT test. Although the error decreases proportionally there is a stabilization point at around 1440 pixels height. This implies that increasing the resolution is beneficial and reduces the distance errors but up to the point of 1440 pixels height. Any increase of the resolution above 1440 pixels height is redundant and will decrease efficiency significantly.

6.3 Degrees added at trajectory

Trajectory Error

Adding 45 – 240 degrees at the trajectory significantly reduces the trajectory error from 5 to 6 meters to 1.5 to 2 meters. This implies that closing the loop and overlapping a few more degrees is beneficial. Now, adding 360 degrees or more increases the error compared to adding 0 – 240, and there is a logical reason for that. The way the degrees were added did not take into consideration that the images could have an overlap. When adding 360 degrees or more it adds a significant overlap which leads to the increase of the error. To properly identify the effects of adding more than 360 degrees, an implementation should take place that will distribute the images in a way to avoid overlap. Nevertheless, the conclusion is that adding 45 – 240 degrees significantly decreases the trajectory error. For more details on what the trajectories looked like and how the validation that a perfect alignment is possible can be found in the Appendices.

6.4 Voxel Size and Bound

Precision and Recall

By increasing the voxel and the bound size the precision of COLMAP is barely affected since it is always very precise and does not reconstruct when it is unsure. On the other hand, the recall at 0.4 m it stabilizes at 71%. This tells us that when setting the voxel and bound size >= 0.4 m all the estimated voxels find a corresponding match. Again, keep in mind that this only stands for the specific experiment with cropped trees of 6-8 meters height.

Distance Error

Regarding the effects on the distance error, for the COLMAP→GT there is a slight increase between 0.2-0.4 m but then it stabilizes in. On the other hand, for the GT→COLMAP there is a more rapid increase in the error. This is because increasing the bound will increase the search space and allow the matching and distance processes to look further thus leading to more matches and bigger errors. Also, the COLMAP→GT stabilizes early at 0.4 m because COLMAP is precise and as mentioned by using size >= 0.4 m all the voxels find a corresponding voxel leading to an unchanged error.

Chapter 7: Conclusions

To summarize, this pipeline documents and automates the basic steps of an evaluation pipeline. This is very useful for two reasons. This pipeline could be used to evaluate new proposed pipelines/methods and more importantly it is the basis to set an automated evaluator to be used during the training process of a learning-based approach. As an evaluator to evaluate reconstruction methods there are a few new additions compared to other evaluation methods. First, it allows the cropping of specific objects from the scene and evaluates only those. This is very useful in the cases where the whole scene is only provided or reconstructed but it is desired to only evaluate specific parts of the scene. This is also beneficial for using it in a learned based approach because it can turn the focus of the reconstruction towards specific parts. Second, it provides visualizations during all the steps that includes the estimated and ground truth PCD, the cropped objects in PCD and voxel grid format, and most importantly the coverage and distance errors of the cropped objects that are color coded. Third, it evaluates the voxel grids. Although the evaluation metrics used are not novel, they are specific for voxel grids that could help the development of the learning-based approach. The results for the experiments show in general that the only improvement that is available regarding COLMAP is to find a way to explore more points and increase the recall. COLMAP, as has been proven from the experiments, is very precise but it cannot detect complex structures leading to missing voxels at the voxel grids and thus a lower recall. Also, the experiments show good practices for increasing the quality and efficiency of the reconstruction process using COLMAP in natural scenes. Briefly, increasing the number of images used during the reconstruction helps by increasing the recall but it is disproportionate. Increasing the resolution does help both in increasing the recall and reducing the distance error but there is a stabilization point at 1440 pixels height. Any resolution higher than 1440 pixels height is unnecessary/redundant and it will only decrease the efficiency. For the degrees added at the trajectory, adding 180 degrees to the trajectory seems to be the sweet spot to reduce the trajectory error. Finally, varying the voxel and bound size plays a role if a size bigger than 0.4m is used. So, the recommended size is 0.4m where the recall stabilizes. Although the results from the experiments show certain guidelines, we do not know how they would perform in different digital scenes or even in real world scenarios.

Chapter 8: Future Work

First, we couldn't find a dataset that contained CAD models of natural environments such as trees, plants etc. This is something that should be considered for future work. Now for the usage of the pipeline as the evaluation step in a learning-based approach a few changes must be made. The pipeline scripts are bonded to the COLMAP pipeline. Thus, a few changes must be made to adapt to the specific learning-based approach. Luckily, some of the steps can be skipped when it is used for the learning-based approach since most of the learning-based approaches do not separate the reconstruction in sparse and dense versions. Second, there are many pathways to approach the learning-based approach, so the next step would be to follow one of the pathways of the learning-based approaches. Some of them have been mentioned in the literature review. The approaches could be to predict voxel grids, point clouds, meshes, or even continuous functions and decision boundaries. We suggest following the voxel grid geometrical representation because of the complexity of the trees, the fine detail characteristics and movement of the leaves and branches, make it very difficult to create an accurate representation. Predicting voxel grids/point clouds is the easiest approach. If accurate voxel grids are predicted for the trees this would still allow the employment of different applications. Not only that, varying the voxel size and trying to first predict using relatively bigger voxel sizes, and reducing them afterwards for fine detail around the search space can still bring sufficient results. More importantly due to the availability of the digital twin this allows us to have a ground truth scene that is accurate thus allows the training of models to predict accurately the point clouds/voxel grids. Another approach would be to create a machine learning evaluator that mimics the observations of the HVS. The works presented for this was [33] but works haven't been found to do that for a point cloud evaluation but only for images. Moreover, a system in [33] could also be trained using only natural scenes so that could also be a future work. Now, we must also consider the scope of the prediction. Will it be for a single tree or a whole natural scene consisting of a number of trees? This decision would play a vital role in the modeling of the machine learning model. It is suggested to use the cropping process to focus on the cropped objects as a first approach. Another more advanced approach could be by following the approach proposed by [6]**,** which learns characteristics of objects and then can reconstruct a mesh from a point cloud. The twist will be that it could be used to learn the details of trees from close views (e.g. details of leaves and branches) and then use these learned characteristics to compose the whole tree. Moreover, a different approach can be implemented like the one proposed in Nerf [5] that the evaluation is solely based on the images. Using digital images will still be helpful but following this approach will avoid the creation of the ground truth scene and other sparse, dense reconstructions that are very computationally expensive. Finally, it is recommended to explore new metrics apart from those implemented here (coverage/matching and distance errors). The distance errors when added (COLMAP → GT, and GT → COLMAP) compose the CD [25] metric. Explore implementing JSD [27] and see how it behaves.

References

[1] J. L. Schonberger and J.-M. Frahm, “Structure-from-Motion Revisited,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 4104–4113. doi: 10.1109/CVPR.2016.445.

[2] J. L. Schönberger, “COLMAP.” Accessed: Dec. 27, 2023. [Online]. Available: https://colmap.github.io/

[3] H. Fan, H. Su, and L. Guibas, “A Point Set Generation Network for 3D Object Reconstruction from a Single Image,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017, pp. 2463–2471. doi: 10.1109/CVPR.2017.264.

[4] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan, “PointFlow: 3D Point Cloud Generation with Continuous Normalizing Flows,” Jun. 2019.

[5] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “NeRF,” *Commun ACM*, vol. 65, no. 1, pp. 99–106, Jan. 2022, doi: 10.1145/3503250.

[6] C. Jiang, A. Sud, A. Makadia, J. Huang, M. NieBner, and T. Funkhouser, “Local Implicit Grid Representations for 3D Scenes,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2020, pp. 6000–6009. doi: 10.1109/CVPR42600.2020.00604.

[7] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *Int J Comput Vis*, vol. 60, no. 2, pp. 91–110, Nov. 2004, doi: 10.1023/B:VISI.0000029664.99615.94.

[8] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-Up Robust Features (SURF),” *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, Jun. 2008, doi: 10.1016/j.cviu.2007.09.014.

[9] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” in *2011 International Conference on Computer Vision*, IEEE, Nov. 2011, pp. 2564–2571. doi: 10.1109/ICCV.2011.6126544.

[10] “Bundle adjustment - Wikipedia.” Accessed: Dec. 28, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Bundle_adjustment

[11] “Computer stereo vision - Wikipedia.” Accessed: Dec. 28, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Computer_stereo_vision

[12] “Binocular disparity - Wikipedia.” Accessed: Dec. 28, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Binocular_disparity

[13] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, “The ball-pivoting algorithm for surface reconstruction,” *IEEE Trans Vis Comput Graph*, vol. 5, no. 4, pp. 349–359, Oct. 1999, doi: 10.1109/2945.817351.

[14] H. Hoppe, “Poisson surface reconstruction and its applications,” in *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, New York, NY, USA: ACM, Jun. 2008, pp. 10–10. doi: 10.1145/1364901.1364904.

[15] “Voxel - Wikipedia.” Accessed: Dec. 28, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Voxel

[16] “Fully Connected Deep Networks - TensorFlow for Deep Learning [Book].” Accessed: Dec. 28, 2023. [Online]. Available: https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html

[17] “Point cloud - Wikipedia.” Accessed: Dec. 28, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Point_cloud

[18] “Polygon mesh - Wikipedia.” Accessed: Dec. 28, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Polygon_mesh

[19] “Lidar - Wikipedia.” Accessed: Dec. 28, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Lidar

[20] “Open Heritage 3D | Data.” Accessed: Dec. 28, 2023. [Online]. Available: https://openheritage3d.org/project.php?id=d51v-fq77

[21] “Photogrammetry - Wikipedia.” Accessed: Dec. 28, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Photogrammetry

[22] V. Sitzmann, J. Thies, F. Heide, M. NieBner, G. Wetzstein, and M. Zollhofer, “DeepVoxels: Learning Persistent 3D Feature Embeddings,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2019, pp. 2432–2441. doi: 10.1109/CVPR.2019.00254.

[23] “Lambertian reflectance - Wikipedia.” Accessed: Dec. 28, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Lambertian_reflectance

[24] A. X. Chang *et al.*, “ShapeNet: An Information-Rich 3D Model Repository,” Dec. 2015.

[25] “Point Cloud Metrics - Point Cloud Utils.” Accessed: Dec. 28, 2023. [Online]. Available: https://www.fwilliams.info/point-cloud-utils/sections/shape_metrics/

[26] Y. Rubner, C. Tomasi, and L. J.Guibas, “The Earth Mover’s Distance as a Metric for Image Retrieval,” *Int J Comput Vis*, vol. 40, no. 2, pp. 99–121, Nov. 2000, doi: 10.1023/A:1026543900054.

[27] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, “Learning Representations and Generative Models for 3D Point Clouds,” 2018. [Online]. Available: http://github.com/optas/latent_3d_points

[28] S. Kullback and R. A. Leibler, “ON INFORMATION AND SUFFICIENCY”, Accessed: Dec. 28, 2023. [Online]. Available: https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-22/issue-1/On-Information-and-Sufficiency/10.1214/aoms/1177729694.full

[29] “Multilayer perceptron - Wikipedia.” Accessed: Dec. 28, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Multilayer_perceptron

[30] “Peak signal-to-noise ratio - Wikipedia.” Accessed: Dec. 28, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio

[31] “Mean squared error - Wikipedia.” Accessed: Dec. 28, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Mean_squared_error

[32] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image Quality Assessment: From Error Visibility to Structural Similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004, doi: 10.1109/TIP.2003.819861.

[33] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The Unreasonable Effectiveness of Deep Features as a Perceptual Metric,” Jan. 2018.

[34] H. R. Sheikh, M. F. Sabir, and A. C. Bovik, “A Statistical Evaluation of Recent Full Reference Image Quality Assessment Algorithms,” *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3440–3451, Nov. 2006, doi: 10.1109/TIP.2006.881959.

[35] N. Ponomarenko *et al.*, “TID2008-A Database for Evaluation of Full-Reference Visual Quality Assessment Metrics.”

[36] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, “Occupancy Networks: Learning 3D Reconstruction in Function Space,” Dec. 2018.

[37] “Root-mean-square deviation - Wikipedia.” Accessed: Dec. 28, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Root-mean-square_deviation

[38] “Introduction to the viridis color maps.” Accessed: Dec. 28, 2023. [Online]. Available: https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html

[39] “TU Delft - Ambulance Drone - YouTube.” Accessed: Dec. 28, 2023. [Online]. Available: https://www.youtube.com/watch?v=y-rEI4bezWc&ab_channel=TUDelft

[40] “Dino - 2019 - Official presentation - Autonomous Mechanical Weeding Robot - YouTube.” Accessed: Dec. 28, 2023. [Online]. Available: https://www.youtube.com/watch?v=fW7UVHz9QYA&ab_channel=Na%C3%AFoTechnologies

[41] “The farming robots that will feed the world | Hard Reset - YouTube.” Accessed: Dec. 28, 2023. [Online]. Available: https://www.youtube.com/watch?v=hBkhUClyJvs&ab_channel=Freethink

[42] “MeshLab.” Accessed: Dec. 28, 2023. [Online]. Available: https://www.meshlab.net/

[43] Q.-Y. Zhou, J. Park, and V. Koltun, “Open3D: A Modern Library for 3D Data Processing,” Jan. 2018, Accessed: Dec. 28, 2023. [Online]. Available: http://arxiv.org/abs/1801.09847

[44] “Online 3D Viewer.” Accessed: Dec. 28, 2023. [Online]. Available: https://3dviewer.net/

Appendices

Appendix A: Emergency drone

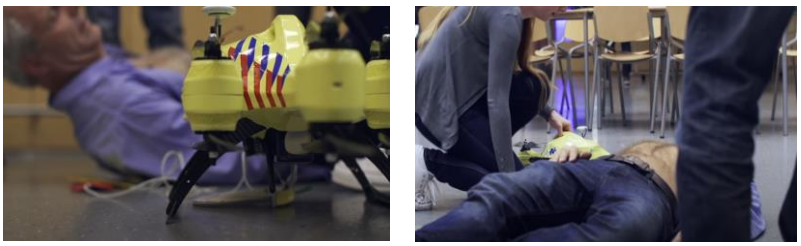


Figure 27: An Emergency Drone [39]

Appendix B: Weeding Robot

Figure 28: Weeding Robot [40]

Appendix C: Picking Robot

Figure 29: Picking Robot [41]

Appendix D: View configuration of scenes

By using the software tool vulkan vr glasses to generate the images of a digital scene the user has a variety of options to change the views of the images that are being captured. One option is the "ground" view, where it simulates an entity going through the scene and taking pictures at a ground level. An example is shown below.

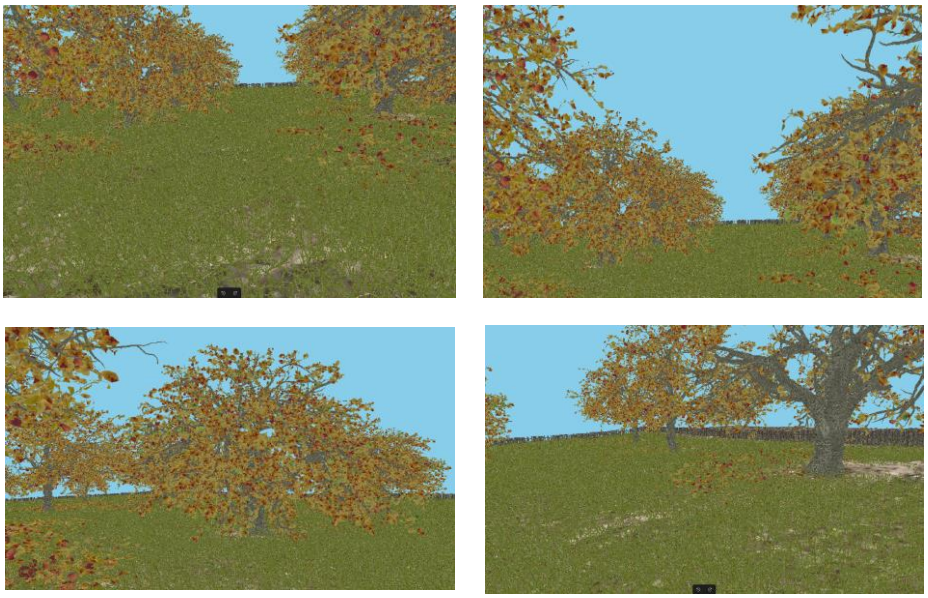


Figure 30: Images of the digital scene with autumn trees captured with ground view

Now, we avoid this view because we are most interested in images that could be captured by drones for obvious reasons such as ease of access and ability to automate the capture process when executed in a real-world environment. This is why we explored the two other views, the "top" view and the "around" view. Below you can see examples of the top view.

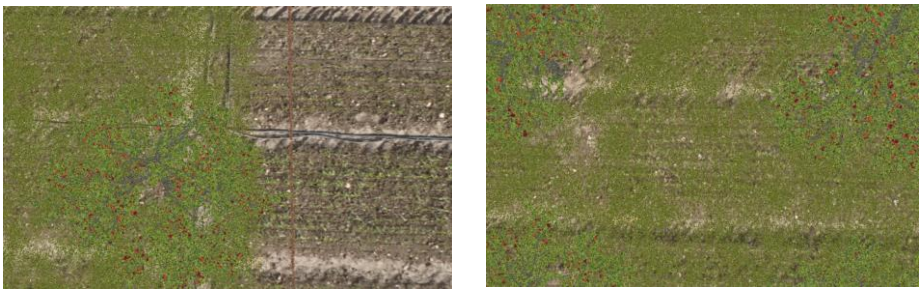


Figure 31: Images of the digital scene with summer trees captured with top view

To explain why the "around" view is the preferable method and why it was chosen for all of the experiments, the sparse reconstruction results after providing images from the "top" view can be seen in Figure 32. A reminder that in the sparse reconstruction the red markers represent the estimated poses of the images captured and the points the estimated points that depict the scene seen in the images.

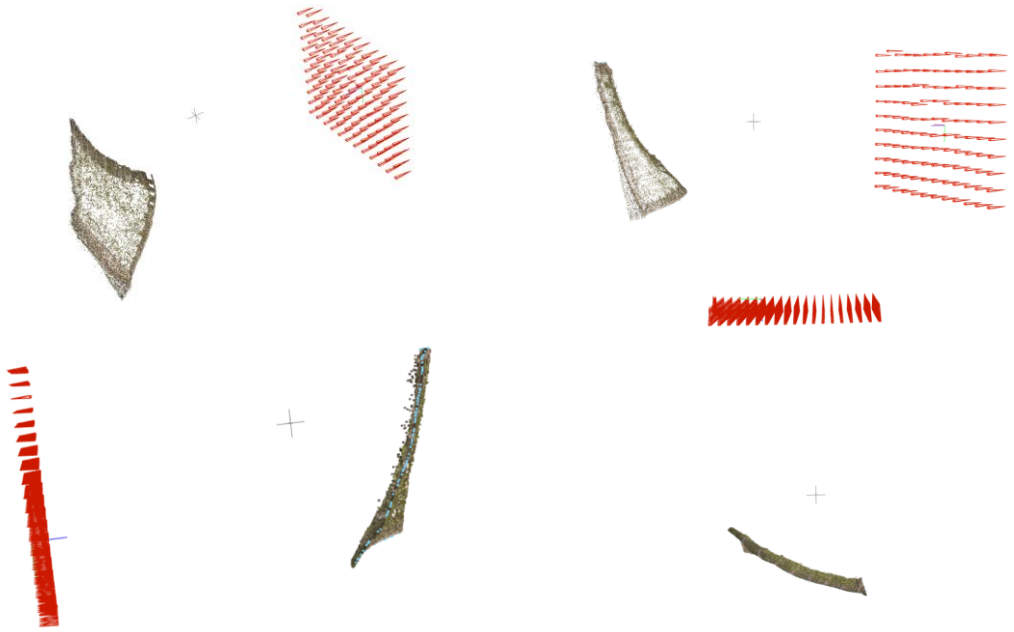


Figure 32: Sparse reconstruction using images captured with top view

Notice the poses of the images that are in a fixed grid and are pointing directly at the scene (this is the top view). Also, notice the curved terrain in all the images and the very sparse point cloud in the center at the images of the top row. In comparison with the around view in Figure 2 the image poses form a circle and have an angle. Also, the terrain in the around view is not curved but is flat as it was supposed to be. Finally, in the around view the appearance of points for the trees are not flatten like in the examples above were no tree points can be seen. For all of the aforementioned reasons this is why we chose the around view as the preferred method as it allows a better quality with less errors. Now, for the degrees that are added at the trajectory, as you can see in Figure 2 there is a single circle of images being taken (see red markers). This is considered to perform a trajectory of 360 degrees. The experiments for the degrees are adding additional degrees to the overall trajectory, so for example if 40 degrees are set to be added it will result to a trajectory that performs 360 + 40 = 400 degrees of a trajectory.

Appendix E: More visualizations of the trajectories

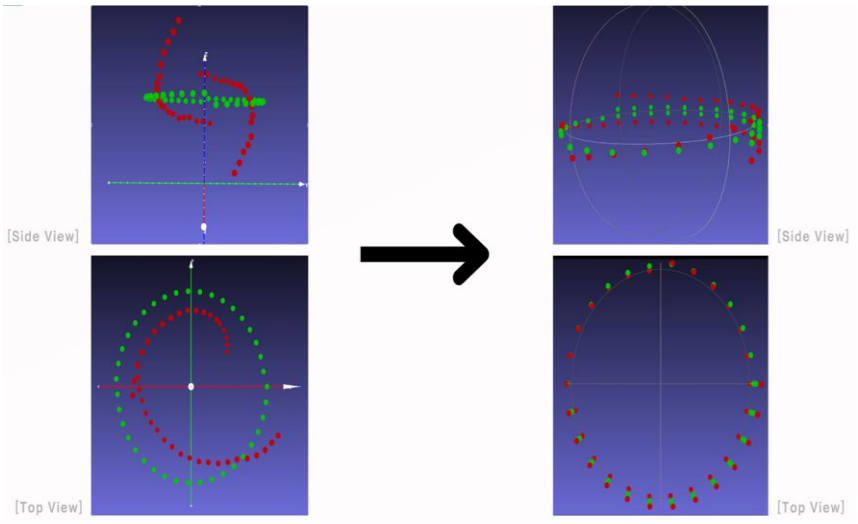
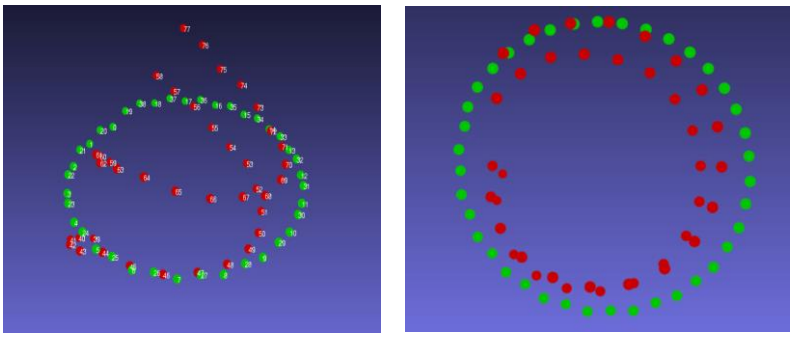


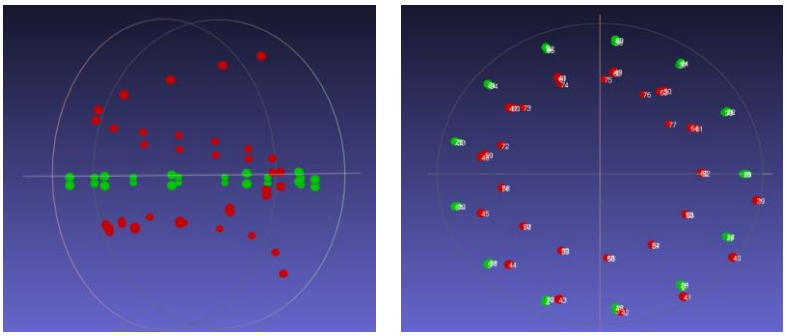
Figure 33: Before (left) and after (right) adding 180 degrees to the trajectory

To understand the cause of the increase of the error some visualizations are provided below.



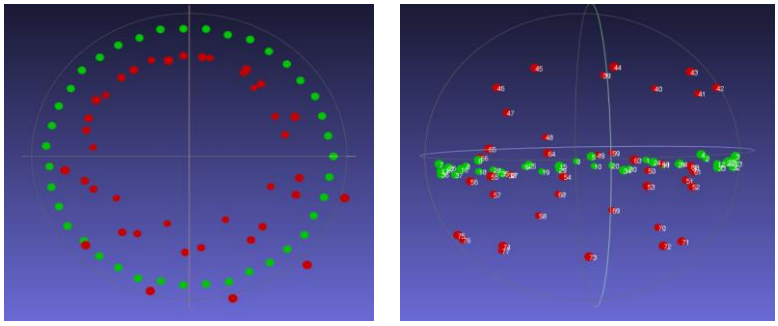
(side view) (top view)

Figure 34: Trajectory when adding 360 degrees



(side view) (top view)

Figure 35: Trajectory when adding 720 degrees



(side view) (top view)

Figure 36: Trajectory when adding 1080 degrees

Appendix F: Verifying the validity of the evaluation process of the image poses

In order to verify that the evaluation of the image poses we have identified an evaluation that perfectly aligns the two trajectories leading to a very low trajectory error. The trajectory error is 0.0003 m.

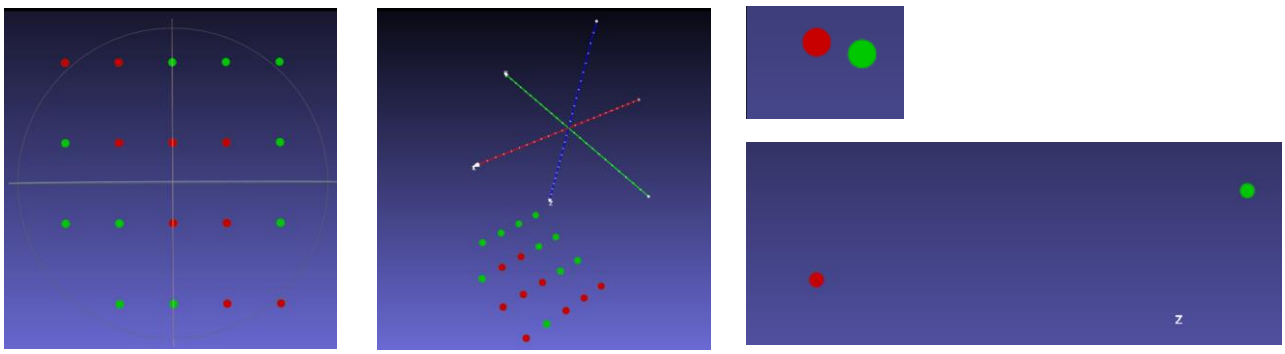


Figure 37: Ideal example of the trajectories to verify the validity

The green color is for the GT and the red color is for the Estimated/COLMAP. Please note that the points are overlapping and by changing the view angle the color would have changed. To understand this we also provide on the right side zoomed in views.

Appendix G: Examples of models

In this section we will be providing a few examples of the models that are being used to reconstruct the digital scene and the ground truth PCD. There are a lot of tools/websites to visualize such models in .obj format such as MeshLab [42], OPEN3D[43], and 3DViewer [44].

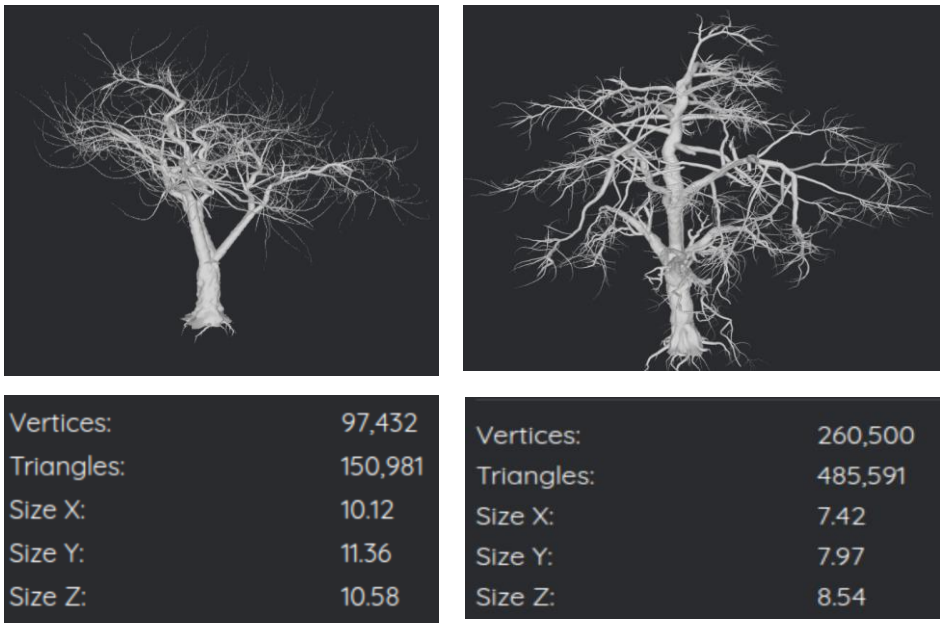


Figure 38: Winter tree objects and their corresponding statistics (left) Apple_Trunk2_dark.obj, (right) Apple_Trunk1_dark.obj Used [44]

Figure 39: Grass object

Figure 40: Fence object

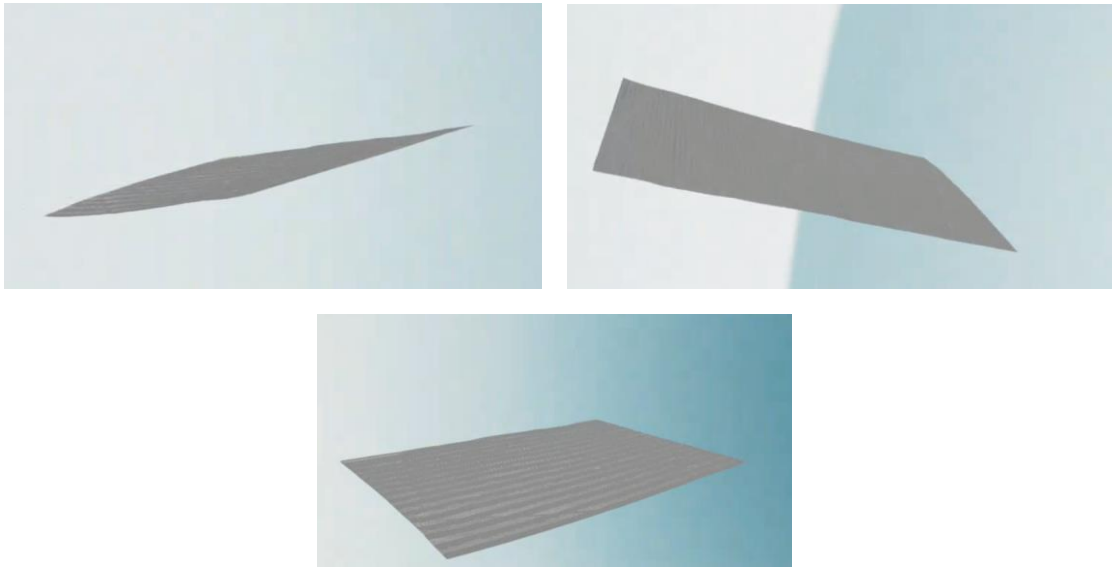


Figure 41: Terrain object

Appendix H: Mendeley
Platform used for to automate the citation process
Available at https://www.mendeley.com/reference-management/mendeley-cite

Appendix I: ChatGPT | Platform used for generic help
Platform used for generic help.
Available at: https://openai.com/chatgpt